REMARKS

The claims are claims 1, 2, 4, 5 and 7 to 13.

Claims 1, 2, 4, 5 and 7 to 13 were rejected under 35 U.S.C. 103(a) as made obvious by the combination of Kling et al U.S. Patent No. 6,883,089 and Yamada et al U.S. Patent No. 6,877,087.

Claims 1 and 4 recite subject matter not made obvious by the combination of Kling et al and Yamada et al. The scoreboard bit recited in claims 1 and 4 is unobvious over the combination of Kling et al and Yamada et al. Claim 1 recites "a scoreboard bit corresponding to each data register capable of serving as a predicate register, each scoreboard bit connected to said instruction decode unit to be set to a first digital state upon determining said corresponding data register is a destination for an instruction and connected to said plurality of functional units to be reset to a second digital state opposite to said first digital state upon functional unit write of a result to said corresponding data register." Claim 4 recites "setting a scoreboard bit to a first digital state upon determining a corresponding data register is a destination for an instruction; resetting a scoreboard bit to a second digital state opposite to said first digital state upon a write of a result to said corresponding data register." The Applicants respectfully submit that the scoreboard bit recited in claims 1 and 4 stores a different indication than the scoreboard bit of Kling et al. Claims 1 and 4 recite placing the scoreboard fit in a first digital state when a write is pending to the corresponding predicate register. This is recited as the "corresponding data register is a destination for an instruction." Claims 1 and 4 recite placing the scoreboard bit in a second digital state upon completion of the pending write. This is recited as "write of a result to said

corresponding data register." The FINAL REJECTION states at page 4, lines 2 to 11:

> "each scoreboard bit connected to said instruction decode unit (each component in a processor is directly or indirectly connected) to be set to a first digital state (not available) upon determining said corresponding data register is a destination for an instruction (when the data register is a destination of an instruction, the register would be unavailable and therefore the scoreboard bit must be set) and connected to said plurality of functional units to be reset to a second digital state opposite to said first digital state (available) upon functional unit write of a result to said corresponding data register (when the instruction is completed, the register becomes available and the scoreboard must be set. If the score is not set, the processor would stall indefinitely);"

The FINAL REJECTION thus states that data is unavailable when a register is the destination of a write. However, the FINAL REJECTION fails to point out any part of Kling et al supporting this assertion. Likewise, the FINAL REJECTION states that the register becomes available when the write instruction completes. Again the FINAL REJECTION points out no part of Kling et al as teaching this subject matter. The scoreboard of Kling et al indicates whether the operands and data in a predicate register are available. Kling states at column 3, lines 16 to 32:

> "A scoreboard 170 indicates if the operands are ready. If any one of the source operands are not available the pipeline stalls. If an instruction is predicated (i.e. has a guarding predicate), the scoreboard 170 is checked next for the availability of the guarding predicate. If the guarding predicate is available, the guarding predicate is read from the register file 168 location containing the guarding predicate. If the guarding predicate is not available a simple machine would stall the pipeline. An enhanced machine of one embodiment could postpone stalling due to an unavailable guarding predicate until the completion of the execute stage 164. The execute stage 164 is the last chance of preventing a predicated-off result to be sent to following instructions via

> a result bypass network. In either case, a 'false' guarding predicate causes the result of the instruction to be discarded while a 'true' predicate allows the result to be written to the register file 168 in the write back stage 166."

Kling et al fails to state what makes the operands or the data in the predicate register available or not available. Kling et al fails to teach placing the scoreboard bit in a first digital state upon detection of a write to a corresponding predicate register and placing the scoreboard bit in a second digital state upon write of a result to the corresponding predicate register. Accordingly, Kling et al fails to teach the scoreboard bit recited in claims 1 and 4 which clearly recite criteria for setting and clearing a scoreboard bit not taught in Kling et al. Thus the FINAL REJECTION makes arguments regarding the teaching of Kling et al not supported by the cited portion of the reference. This appears to be an argument based upon inherency, but the FINAL REJECTION includes no arguments why the recited limitations are inherent in Kling et al. Note that this application teaches at page 4, lines 11 to 14:

> "When a write is pending to the predicate register, the instruction is allowed to execute normally, with the result write-back happening only at a later stage in execution dependent upon the newly written predicate value."

In accordance with the recitations of claims 1 and 4, the scoreboard bit is set "When a write is pending to the predicate register." This portion of the application states that the instruction is allowed to execute normally. This is in contrast to the teachings of Kling et al which require the execution to stall until the scoreboard indicates the predicate register is available. This difference is based upon the fact that the scoreboard bit claimed in claims 1 and 4 indicates different status than scoreboard 170 of Kling et al. Kling et al includes no details on what would make an operand or predicate unavailable. Only this

application teaches setting a scoreboard bit when a write is pending to a predicate register. Thus the Examiner's inherency argument is based upon the teachings of this application and no upon the teachings of Kling et al. This is improper use of the teachings of this application. The FINAL REJECTION does not allege that Yamada et al makes obvious this subject matter. Accordingly, claims 1 and 4 are unobvious over the combination of Kling et al and Yamada et al.

The FINAL REJECTION states at page 11, line 18 to page 12, line 5 and at page 12, lines 13 to 22:

> "Inherently, if an instruction is in a pipeline, the result register is clearly unavailable because the instruction writing to that register is not completed. Also, when the instruction is complete, the register must inherently become available (otherwise the processor would halt and progress could not be made). It is impossible for data to be available if an instruction which is not completed targets that register (since the data for that register is stale and must be updated). Once the instruction executes, the data is then available. Since data registers must be targeted in any processor and inherently, if a register is targeted, it is unavailable for at least some amount of time, the system of Kling teaches setting the bit in response to a "write of a result to said corresponding data register" (because it is now unavailable)."

The Applicants respectfully submit that this inherency argument is incorrect. It is well known in the data processing art that an instruction can write its results to a register serving as one of its input operands. Kling et al includes an example of such an instruction in Figure 2. Instruction 4 of Figure 2 of Kling et al is:

```
"(P1)      add 5, R1 -> R1      # adds 5 to R1 if P1 is true"
```

If Kling et al operated according to the Examiner's inherency argument, this instruction could never execute. The register R1

could not be read for the input operand because this register is the destination of the instruction and thus unavailable according to the Examiner's inherency argument. It is known in the art that such an instruction operates by reading the input operand register before the results are stored into that register. This example instruction contradicts the Examiner's supposed inherent operation of Kling et al. The Applicants respectfully submit that Kling et al would not include an example instruction that inherently would not operate. Accordingly, the Examiner's argument regarding the inherent operation of Kling et al is incorrect. This application teaches the first instruction in execute packet 8 of the code example (page 21, line 20):

"[A2] SUB .D1 A2, 1, A2"

This instruction uses function unit D1 to subtract 1 from the contents of register A2 and store the results in register A2 predicated by value stored in register A2. Thus in the context of this application the fact that a write is pending to a register does not make that register unavailable for use as an input operand or as a predicate. According to the Examiner's inherency argument, such instructions could never execute because the scoreboard would stall execution of the instruction waiting for the write of that instruction to complete. Since such instructions are taught both in Kling et al and in this application, this is not inherent in Kling et al. The Applicants respectfully submit that this limitation is not made obvious by the combination of Kling et al and Yamada et al and is not inherent in the operation of Kling et al. Accordingly, claims 1 and 4 are allowable over the combination of Kling et al and Yamada et al.

Claims 1 and 4 recite subject matter not made obvious by the combination of Kling et al and Yamada et al. Claims 1 and 4 recite

operating at a reduced power state under conditions not obvious from the combination of Kling et al and Yamada et al. Claim 1 recites "each functional unit is further operative responsive to a predicate instruction during said instruction decode pipeline phase to nullify said predicate instruction of a following execution phase by operating at a reduced power state relative to normal instruction operation if said predicate register has said second state during said instruction decode pipeline phase and said corresponding scoreboard bit has said second state during said instruction decode pipeline phase." Claim 4 similarly recites "nullifying a predicate instruction for a following execution phase by operating said corresponding functional unit at a reduced power state relative to normal instruction operation if said corresponding predicate register has said second state during a prior instruction decode pipeline phase and said corresponding scoreboard bit has said second state during a prior instruction decode pipeline phase." The FINAL REJECTION admits that Kling et al fails to teach this subject matter. The FINAL REJECTION cites Yamada et al at column 5, lines 46 to 54 as allegedly making this limitation obvious. This combination of Kling et al and Yamada et al fail to teach this limitation for three reasons.

Firstly, claims 1 and 4 recite a contingency not determined by Kling et al. Claims 1 and 4 recite the "if said predicate register has said second state during said instruction decode pipeline phase and said corresponding scoreboard bit has said second state during said instruction decode pipeline phase." Kling et al never considers the data in the predicate register and scoreboard bit together. In Kling et al the pipeline stalls an instruction if the scoreboard bit indicates an operand or the predicate register is unavailable during the instruction execution phase. Such a stall is independent of the state of the predicate register. In Kling et al the predicated instruction writes the result to a register or

re-order buffer if the predicate register has a first state and fails to write the result if the predicate register has a second state. Thus Kling et al never conditions any operation on the status of both the predicate register and the scoreboard bit as recited in claims 1 and 4.

The FINAL REJECTION states at page 13, lines 4 to 8:

> "Secondly, if the processor of Kling stalls the pipeline if the scoreboard bit is a second state, then using simple logic, inherently, it must always stall when a more exclusive state occurs (both the scoreboard is a second state and the predicate is a second state). When "C if A and (B or not B)" then inherently, "C if A and B" is inherently true."

Claims 1 and 4 do not recite stalling the pipeline as Kling et al does in this argument of the Examiner. Claims 1 and 4 recite nullifying a "predicate instruction of a following execution phase by operating at a reduced power state relative to normal instruction operation." According to this recitation, the instruction operation is nullified due to the status of the predicate register and the functional unit operates at reduced power for this execution phase. This recitation is parallel in language to the normal predicate operation of nullifying a predicate instruction by not writing the result as also taught in Kling et al. In either case the instruction is not stalled but completes during the nullified execution phase. The stall taught in Kling et al fails to complete the instruction. The data processor waits until the scoreboard bit indicates that all operands and the predicate are available. Only at that time does the instruction complete either calculating and writing the result, or calculating and not writing the result depending on the predicate value. Thus whether or not the data processor operates at reduced power during the stall taught in Kling et al, there is

no teaching of other than normal, full power operation during the execution phase following the stall.

Secondly, the reduced power operation of Yamada et al is based upon a static determination of the instruction stream and does not take into account data calculated at run time. The portion of Yamada et al cited in the FINAL REJECTION teaches reduced power operation in input latch 15, ALU 16 and output latch 17 upon detection of a NOP instruction. Such a NOP instruction must have been placed in the instruction stream upon instruction generation. Yamada et al also teaches at column 6, lines 34 to 59 placing floating-point input latch 22, floating-point data path 23 and floating-point output latch 24 in a low power state by outputting a NOP instruction from invalidation logic circuit 9 upon detection of an instruction operating only on ALU 16 and not operating on floating-point data path 23. This reduced power operation is also completely determined by the instruction stream upon instruction generation. This application teaches at page 27, lines 25 and 26:

> "This invention uses the dynamic information not available to the compiler to control instruction execution."

In contrast, the power reduction technique of Yamada et al uses only information available to the compiler and does not take into account information only available at run time. Accordingly, one skilled in the art would not be motivated to employ the statically determined reduced power operation of Yamada et al with a dynamic system responsive to data values occurring while the system operates such as Kling et al.

Thirdly, the condition determination recited in claims 1 and 4 occurs at a different time than taught in Kling et al. The above quoted portions of claims 1 and 4 recite actions taken based upon certain conditions "during said instruction decode pipeline phase." The determination to distribute or discard the results of an

instruction as determined by the predicate in Kling et al always occurs at the end of the execution pipeline phase of the dependent instruction. Since Yamada et al fails to teach any such data dependent determination, the combination of Kling et al and Yamada et al fail to make obvious claims 1 and 4.

The FINAL REJECTION states at page 13, lines 13 and 14:

> "Lastly, the claim merely states that the instruction is in a decode stage and not that the determination occurs at that time."

Claims 1 and 4 do not recite that the determination takes place during a decode phase. Claim 1 recites "if said predicate register has said second state during said instruction decode pipeline phase and said corresponding scoreboard bit has said second state during said instruction decode pipeline phase." Claim 4 similarly recites "if said corresponding predicate register has said second state during a prior instruction decode pipeline phase and said corresponding scoreboard bit has said second state during a prior instruction decode pipeline phase." Thus this limitation is not upon when the nullify decision is made but upon when the status of the scoreboard bit and the predicate register are determined. Kling et al teaches determining the state of the scoreboard bit at the beginning of the execute phase and writing or not writing the result based upon the status of the predicate during the execute phase. Thus Kling et al teaches testing the status of these quantities at a different time than recited in claims 1 and 4. Accordingly, claims 1 and 4 are allowable over the combination of Kling et al and Yamada et al.

Claims 1 and 4 recite further subject matter not made obvious by the combination of Kling et al and Yamada et al. Claims 1 and 4 recite actions taken based upon the state of the predicate register independent of the state of the scoreboard bit. Claim 1 recites "responsive to a predicate instruction to write said result to an

instruction designated destination data register if said corresponding predicate data register has a first state during said execution pipeline phase regardless of said state of said corresponding scoreboard bit and to nullify said instruction and not write said result if said predicate register has a second state opposite to said first state during said execution pipeline phase regardless of said state of said corresponding scoreboard bit." Claim 4 recites "performing a data processing operation via a corresponding functional unit on at least one source operand recalled from at least one corresponding instruction designated source data register and producing a result in response to a predicate instruction designating a corresponding predicate data register and writing said result to an instruction designated destination data register regardless of said state of said corresponding scoreboard bit if said corresponding predicate data register has a first state during said execution pipeline phase" and "nullifying a data processing operation of a predicate instruction by not writing said result to the instruction designated destination data register via said corresponding functional unit regardless of said state of said corresponding scoreboard bit if said corresponding predicate register has a second state opposite to said first state during said execution pipeline phase." As demonstrated in the continuation of paragraph 11 in the ADVISORY ACTION of December 20, 2006, all operations disclosed in Kling et al are dependent upon the scoreboard bit indicating that data in the predicate register is available. In particular, Kling et al always teaches that scoreboard 170 must indicate that the predicate register is available for the write to the register file to abort. This fails to teach nullifying an operation regardless of the state of the corresponding scoreboard bit as recited in claims 1 and 4. The reason that this invention can operate regardless of the state of the scoreboard bit under

certain conditions is that the scoreboard bit recited in claims 1 and 4 stores a different indication than scoreboard 170 taught in Kling et al. The FINAL REJECTION does not allege that Yamada et al makes obvious this limitation. Accordingly, claims 1 and 4 are not made obvious by the combination of Kling et al and Yamada et al.

The FINAL REJECTION states at page 13, lines 15 to 19:

> "Regarding the argument direct to the term 'regardless', Examiner disagrees. As stated in the previous Office Action, if the instruction is in the execution pipeline phase, the scoreboard bit is already known and therefore the decision whether or not to nullify the instruction is not based on the scoreboard bit and it made regardless of the scoreboard bit."

The Examiner's argument is incorrect. Assuming that the instruction is in the execution pipeline phase ignores the teaching of Kling et al that the data processor stalls unless the scoreboard bit indicates the operands and predicate are available. Accordingly, the decision whether or not to nullify the instruction can only be made if the scoreboard bit indicates the operands and predicate are available. Stated differently, this decision is never reached because the pipeline stalls if the scoreboard bit indicates operands or predicate are unavailable. Thus this operation of Kling et al only happens for a particular state of the predicate bit, that is, an indication that operands and predicate are available. The Applicants submit that this contingency taught in Kling et al is thus not "regardless of said state of said corresponding scoreboard bit" as recited in claims 1 and 4 but in fact dependent upon the status of the scoreboard bit. Accordingly, claims 1 and 4 are allowable over the combination of Kling et al and Yamada et al.

Claims 2 and 5 recite subject matter not made obvious by the combination of Kling et al and Yamada et al. Claims 2 and 5 recite resetting "a scoreboard bit to a second digital state upon

nullification of said instruction designating said corresponding data register as a destination operand data register." Regarding claim 2 the FINAL REJECTION states at page 6, lines 7 to 9:

> "*Note that the processor must update the scoreboard in response to nullifying an instruction. If the scoreboard is not updated, the processor could stall indefinitely waiting for operands to become available.*"

This appears to be an argument based upon inherency from Kling et al. Kling et al fails to teach that upon nullifying "an instruction writing to the predicate bit" "the scoreboard MUST be updated to allow execution of dependent instructions." Kling et al teaches that scoreboard 170 indicates whether the operands and the guarding predicate are available. Kling et al fails to teach what conditions would make the guarding predicate unavailable. Accordingly, Kling et al does not teach that nullifying an instruction writing to a predicate register changes the status of the scoreboard bit. Kling et al only teaches that the scoreboard bit indicates availability of the operands and the guarding predicate. Thus Kling et al fails to teach the operation recited in claims 2 and 5.

In addition, claims 2 and 5 do not require this change of the scoreboard bit supposedly inherent in Kling et al to operate properly. As taught in the application at page 23, line 13 to page 6, line 8, the scoreboard bit is set upon detection of a write to the corresponding predicate register (page 23, lines 15 to 21) and reset upon either a commit of that write (page 23, lines 24 to 29) or a nullification of that write (page 23, line 29 to page 24, line 5). This application teaches that only an early nullification of the dependent instruction and operation at reduced power depends on the scoreboard bit. This application teaches at page 25, line 20 to page 26, line 4 that if the early nullification decision cannot

be made the instruction proceeds with the regular nullification decision. Base claims 1 and 4 recite normal instruction operation if the instruction is not a predicate instruction. Base claims 1 and 4 recite predicated instruction operation during an execute pipeline phase based upon the value of the predicate register "regardless of said state of said corresponding scoreboard bit." Thus the stall of Kling et al noted by the Examiner is not required in this application because this application does not teaches the operation stalls based upon the state of the scoreboard bit. This difference in operation between Kling et al and this invention as recited in claims 2 and 5 occurs because the scoreboard bit of this application differs from the scoreboard bit of Kling et al and is used for a different purpose. Because the scoreboard bit of this application differs from the scoreboard bit of Kling et al the resetting behavior differs. The FINAL REJECTION does not allege that any part of Yamada et al makes obvious this subject matter. Accordingly, claims 2 and 5 are allowable over the combination of Kling et al and Yamada et al.

The FINAL REJECTION states at page 14, lines 1 to 11:

> "Regarding the argument directed to the limitation 'a scoreboard bit to a second digital state upon nullification of said instruction designating said corresponding data register as a destination operand data register', Examiner disagrees. As stated in the previous Office Action, if a nullified instruction would not update the scoreboard, the processor would stall indefinitely because the scoreboard would not allow any dependent instructions to execute. Therefore, inherently, if any instruction leaves the pipeline (is nullified or completes), it must update the scoreboard. Additionally, Examiner is not arguing that the nullification is inherent, but if a nullification occurs, the scoreboard MUST be updated. Applicant's invention shows this in fig. 5. Nowhere in the specification does it state that the scoreboard bit is not updated in some cases when instructions are nullified."

This argument of the Examiner is based upon the incorrect theory that a pending write to an operand or predicate register makes their data unavailable. As noted above, Kling et al includes an example instruction that writes its results to an input operand register. The existence of such an example in Kling et al negates any inference that a pending write to a register makes that register unavailable for that instruction. Since Kling et al fails to teach that a pending write to a register makes its contents unavailable, it likewise fails to teach that nullification of such a pending write makes it available. Accordingly, Kling et al fails to inherently require resetting the scoreboard bit when a pending write is nullified. Accordingly, claims 2 and 5 are allowable over the combination of Kling et al and Yamada et al.

Claim 7 recites subject matter not made obvious by the combination of Kling et al and Yamada et al. Claim 7 recites "scheduling via said compiler a last write to a data register before an instruction decode pipeline phase of a predicate instruction designating said data register as a predicate register." The FINAL REJECTION notes instructions 3 and 4 illustrated in Figure 2 of Kling et al as teaching this limitation. Figure 2 of Kling et al illustrates instruction 3 (cmp 0, R2 -> P1) writing to the predicate register P1 immediately prior to instruction 4 ([P1] add 5, R1 -> R1) guarded by the predicate. In accordance with the teaching of both this application and Kling et al the "last write" to the predicate data register (instruction 3) thus occurs during the instruction decode pipeline phase of the use instruction (instruction 4). Kling et al fails to teach that any change to the predicate data register such as his instruction 3 must be scheduled before the decode phase of his predicate instruction 4 as recited in claim 7. The cited example in Kling et al teaches the last write to the predicate register P1 during the instruction decode pipeline phase of the use instruction rather

than before this instruction decode pipeline phase as recited in claim 7. The FINAL REJECTION does not allege that any part of Yamada et al makes obvious this subject matter. Accordingly, claim 7 is allowable over the combination of Kling et al and Yamada et al.

The FINAL REJECTION states at page 14, lines 12 and 13:

> "Regarding claim 7, scheduling is done with a compiler which is done prior to execution."

Claim 7 does not recite scheduling before execution. Claim 7 recites that instructions are scheduled so that "a last write to a data register before an instruction decode pipeline phase of a predicate instruction designating said data register as a predicate register." This limitation is upon the placement within the instruction stream of the last write instruction and the predicated instruction. This limitation requires this last write to be scheduled to occur before the instruction decode pipeline phase of the predicated instruction. The Applicants respectfully submit that no combination of the teachings of Kling el al and Yamada et al makes obvious this scheduling.

Claims 12 recites subject matter not made obvious by the combination of Kling et al and Yamada et al. Claim 12 recites an OR gate with "a first input receiving a signal from a corresponding functional unit indicating when a write instruction to said corresponding predicate register commits, a second input receiving a signal from said corresponding functional unit indicating said write instruction to said corresponding predicate register nullifies" and a flip-flop having "a set input receiving an input from said instruction decode unit indicating when said corresponding predicate register is a destination for said write instruction, a reset input connected to said output of said OR

gate." The FINAL REJECTION states at page 9, lines 9 to 13, referring to the recited OR gate and flip-flop:

> "All of these components were all very common and well known in the art at the time of the invention. These components are mere functional equivalents to anything else that would perform the same actions (see above regarding claim 1). Using these • components in the combined invention of Kling and Yamada would have been functionally equivalent and would have yielded predictable results."

Using a flip-flop to embody the scoreboard bit is obvious from Kling et al. However the remainder of this argument is incorrect. This argument would be correct if Kling et al included any indication of the three signals recited in claim 12. Kling et al fails to include any teaching of the three signals recited: (1) write instruction commits; (2) write instruction if nullified; and (3) predicate register is the destination of a write instruction. The FINAL REJECTION includes no citation to any part of Kling et al or to Yamada et al as allegedly making these signals obvious. The FINAL REJECTION includes arguments that these signals are inherent in Kling et al. However, these arguments are not based upon any cited teachings of Kling et al. The Applicants respectfully submit that a pending write to a register does not make its contents unavailable as argued by the Examiner. If the contents of a register are read before completion of a pending write, the prior value before the pending update is read. The existence of instructions that write results to an input operand register (as shown in instruction 4 of Figure 2 of Kling et al) negates any inference that a pending write to a register makes its prior data unavailable. Since Kling et al fails to teach the recited signals, it fails to make obvious the recited scoreboard bit construction. Accordingly, claim 12 is allowable over the combination of Kling et al and Yamada et al.

Claim 13 recites subject matter not made obvious by the combination of Kling et al and Yamada et al. Claim 13 recites a first AND gate receiving three inputs. One of these inputs is "a signal indicating when said predicated instruction is in said instruction decode pipeline phase." The FINAL REJECTION states at page 11, lines 3 to 7:

> "All of these components were all very common and well known in the art at the time of the invention. These components are mere functional equivalents to anything else that would perform the same actions (see above regarding claim 1). Using these components in the combined invention of Kling and Yamada would have been functionally equivalent and would have yielded predictable results."

This argument would have merit if Kling et al included any teaching of the claimed signals or their use. The Applicants respectfully submit that no portion of Kling et al teaches the above quoted "a signal indicating when said predicated instruction is in said instruction decode pipeline phase." The FINAL REJECTION includes no argument why this signal would be obvious from the combination of Kling et al and Yamada et al. Accordingly, claim 13 is allowable over the combination of Kling et al and Yamada et al.

Claims 8 to 11 are allowable by dependence upon respective allowable base claims 1 and 4.

The Applicants respectfully submit that all the present claims are allowable for the reasons set forth above. Therefore early entry of this amendment, reconsideration and advance to issue are respectfully requested.

If the Examiner has any questions or other correspondence regarding this application, Applicants request that the Examiner contact Applicants' attorney at the below listed telephone number and address to facilitate prosecution.

Texas Instruments Incorporated
P.O. Box 655474 M/S 3999
Dallas, Texas 75265
(972) 917-5290
Fax: (972) 917-4418

Respectfully submitted,

/Robert D. Marshall, Jr./
Robert D. Marshall, Jr.
Reg. No. 28,527